EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Langer (reg. 30,564) on 11/16/2009.

The application has been amended as follows:

In the specification:

The title has been changed to: Method, a language and a system for the definition and implementation of software solutions by using a visualizable computer executable modeling language.

In page 1, line 14, "Visual Basic" has been changed to –Visual Basic™--.

In page 1, line 15, "Java" has been changed to –Java™--.

In the Claims:

Claim 1. (currently amended) A modeling method for defining software applications by a developer using a visualizable computer executable modeling language, said method comprising:

providing a plurality of display elements for displaying screen objects on a display screen;

displaying components corresponding to said screen objects[[;]] to create an application model using the modeling language by:

defining each of the software applications as a hierarchy of process models, input and output slots, data models, and flow rules[[;]], wherein the components and screen objects are used for defining each of the software applications, the defining of each of the software applications further comprising:

classifying some of said process models and said data models that must not contain any sub-process as atomic;

classifying all other process models and data models that contain one or more sub processes as composite;

defining each of said composite process models as a construction of ~~at least one of~~ one or more sub process models, said input and output slots, said data models, and said flow rules;

classifying inputs sent to the input slots into mandatory and optional;

defining a portion of said input slots ~~of~~ as ~~having~~ being one of ~~the following sub-classifications:~~ mandatory[[;]] and optional, wherein a process that requires an input at each of its mandatory input slots for its initiation, does not start before all said mandatory inputs are received;

defining each of said composite data models as a construction of ~~at least~~ one or more sub data models; [[and]]

defining each of said flow rules [[as]] by connecting a pair selected from ~~of~~ said input and output slots, said data models, and the one or more sub data models, wherein said flow rules define both data flow and process flow in the process and data models, wherein said process models, [[and]] data models, [[and]] input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules[[,]];

~~such that once the software application is visually defined on a display screen by the developer using the modeling language, an application model has been created,~~

thus creating the application model as visually defined on the display screen by the developer using the modeling language; and

enabling a computer to automatically execute the application model defined in said modeling language without requiring further coding according to the definition of the process models, data models, input and output slots, and flow rules arranged in the hierarchy.

Claim **2**. (currently amended) [[A]] The method of claim 1, operating a visualizable computer executable modeling language system ~~operating in accordance with the method of claim 1,~~ for a complete definition of the software applications, said system comprising:

process models, ~~each~~ some of which [[may]] ~~contains~~ contain any number of sub process models, input and output slots, data models and flow rules;

data models, ~~each~~ some of which [[may]] ~~contains~~ contain any number of sub data models; and

flow rules, each of which connecting a pair of said input and output slots, data models and sub data models, thereby defining said data flows and process flows.

Claim **3**. (Currently Amended) The method of claim 2, wherein the modeling language system ~~of claim 2,~~ further ~~comprising~~ comprises at least one visual representation.

Claim **4**. (Currently Amended) The method of claim 3 ~~modeling language system of claim 3~~, wherein said visual representation comprises:

process diagrams comprising various two dimensional shapes representing said process models;

sub process diagrams comprising various two dimensional shapes contained within said process diagrams, representing said sub process models;

slot diagrams comprising various two dimensional shapes situated on the edges of said process diagrams and said sub process diagrams, representing said input and output slots;

data trees comprising hierarchical tree structures contained within said process diagrams, representing said data models and said sub data models; and

flow arrows comprising arrows connecting pairs of said slot diagrams, said data trees and sub-trees of said data tress, said arrows representing said flow rules.

Claim **5**. (currently amended) The method ~~modeling language system of~~ claim 2, ~~wherein each of said slots is further defined as having one of the following classifications: input slot; and output slot (exit); and~~ further defining each of said input slots as having one of the following sub-classifications: synchronous input slot (trigger); and asynchronous input slot; and further defining some of said output slots (exits) [[exit]] as having the sub-classification terminating.

Claim **6**. (Currently Amended) The method ~~modeling language system of~~ claim **2**, ~~wherein each of said slots is further defined as having one of the following classifications:~~

~~input slot; and~~

~~output slot (exit);~~

~~and~~ further defining each of said flow rules contained in said composite process models as connecting one source and one target;

and further defining the source of each of said flow rules to be one of the following:

an input slot of said composite process models; an output slot (exit) [[exit]] of a sub process model of said composite process models;

a data model of said composite process models; and

a sub data model of a data model of said composite process models;

and further defining the target of each of said flow rules to be one of the following:

an exit of said composite process models;

an input slot of a sub process model of said composite process models;

a data model of said composite process models; and

a sub data model of a data model of said composite process models.

Claim 7. (Currently Amended) The method ~~modeling language system~~ of claim 2, wherein:

~~each~~ some of said process models [[may]] further ~~contains~~ contain a reference to a database table (process table);

at least some of the sub data models of data models of said process models are marked as interesting fields; and

each of said interesting fields further contains a reference to a column of said process table.

Claim 8. (Currently Amended) The method ~~modeling language system~~ of claim 7, wherein:

a selection condition of an SQL query (addressing clause) ~~may be~~ can be attached to an input slot of said process models to select matching instances of said process models each time data is to be received by said instances through said input slot;

said addressing clause is defined in terms of a matching condition between the interesting fields of said process models and the data models of said data to be received through said input slot.

Claim 9. (Currently Amended) The method ~~modeling language system~~ of claim 2, wherein ~~each~~

some of said process models [[may]] further ~~contains~~ contain a reference to a computer code implementing the function of said process models.

Claim **10**. (Currently Amended) The method ~~modeling language system~~ of claim **2**, wherein:

each of said composite data models is composed of said sub data models by one of the following structure means: concatenation; collection; and selection, each of said sub data models having a classification as one of the following: mandatory; and optional; each of said sub data models [[may]] can be further [[be]] marked as recurring, with a further optional indication of minimal and maximal number of occurrences;

each of said data models [[may]] can further contains constraints on the data it defines, comprising at least one of the following: legal characters; and minimal and maximal length; each of said data models [[may]] can further comprises a set of legal values and an initial value; and

each of said data models [[may]] can further comprises formatting directives.

Claim **11**. (Currently Amended) [[A]] The method of claim 2 operating a modeling system for defining the software applications using the visualizable computer executable modeling language system ~~of claim **2**~~, enabling users to create, display, modify and test, in an integrated workspace, models of said modeling language, in accordance with the rules of said modeling language system, wherein[[:]] said modeling system comprises a graphical user interface tool (visual modeling tool) for creating, displaying, modifying and testing models of said modeling language in an integrated

workspace, such that users of said modeling tool create and edit said models using various graphical user interface (GUI) operations.

Claim **12**. (Currently Amended) The method ~~modeling system~~ of claim **11**, wherein each of said process models and said data models is further defined as having one of the following classifications: dependent model[[:]] which only exists as a sub model of a specific parent model; and reusable model[[:]] which [[may]] can be reused as a sub model of multiple parent models, wherein each of said reusable models is assigned a unique identifier.

Claim **13**. (Currently Amended) The method ~~modeling system~~ of claim **11**, wherein models are formally represented as one of the following:

Extensible Markup Language (XML) documents;

structured database records; and any other equivalent binary representation, and wherein a repository of said representations of said models, arranged as a hierarchy of packages and sub-packages (knowledge base), is used to maintain libraries of said models, and wherein the modeling system displays said models whose said representations are stored in said knowledge base, stores in said knowledge base said representations of new said models that are defined by the users of the modeling system, and updates said representations of said models in said knowledge base according to modifications made to said models by said users.

Claim **14**. (Currently Amended) The method ~~modeling system~~ of claim **11**, further comprising at

least the following editing capabilities: selection of editing operations from menus; adding components to said models through dragging of models from palettes of existing models; and modifying attributes of said models and components of said models.

Claim **15**. (Currently Amended) The method ~~modeling system~~ of claim **11**, wherein said workspace comprises a drawing board for displaying hierarchies of two dimensional visual diagrams, each representing a corresponding said hierarchy of models, and wherein said users are able to zoom in and out from a currently displayed part of said hierarchy of diagrams, enabling the display of the details of said model and any sub-model thereof at any desired level of said hierarchy of models.

Claim **16**. (Currently Amended) The method ~~modeling system~~ of claim **11**, further comprising: a software program (runtime engine) to execute models defined in said modeling language; and a visual debugger for testing and debugging said models, wherein: said runtime engine, as it executes said models, produces records listing the details of said execution (trace events); said trace events are used to record and store the history of said execution; and said visual debugger uses said stored trace events to display the current status of instances of processes, including the content of their data, as well as the processing steps that have led to said current status.

Claim **17**. (Currently Amended) The method of claim 2, enabling a runtime engine ~~A software program (runtime engine)~~ to execute the applications defined in the visualizable computer executable modeling language system ~~of claim 2~~, wherein[[:]] each of the applications is defined

by a single said process model and a hierarchy of its sub-models; and the runtime engine executes [[the]]each application exactly as defined by said single process model and said hierarchy of its sub-models, thus eliminating the need for writing computer source code in any programming language to implement [[the]]each application.

Claim **18**. (Currently Amended) The method ~~runtime engine~~ of claim **17**, wherein the runtime engine further employs ~~comprising~~: active models comprising objects responsible for representing and enacting the definitions and rules embodied in said models, where there is an active model corresponding to each of said process models and data models; runtime objects comprising objects containing the runtime state of instances of said process models and data models, where there may be at any time any number of runtime objects instantiated from each of said process models and data models by the corresponding said active model; and a model loader comprising an object responsible for loading said models from their formal representations stored in a repository, converting said loaded models to corresponding said active models, and caching said active models.

Claim **19**. (Currently Amended) The method ~~runtime engine~~ of claim **17**, wherein the runtime engine executes each of said models as a series of processing steps, wherein: each of said processing steps is triggered by the receipt of an external input; the runtime engine invokes at least one instance of at least one relevant process model to handle said received input, and executes sub-processes of said invoked processes as defined by the relevant said flow rules; and

a processing step ends when any further activities to be performed depend on the receipt of other external inputs.

Claim **20**. (Currently Amended) The method ~~runtime engine~~ of claim **17**, wherein[[:]] the runtime engine executes each of said models as a series of processing steps by invoking at least one instance of said process models; the full state of each of said at least one process instance is made persistent at the end of each said-processing step; execution of each of said at least one process instance can resume from its stored state at any relevant time; and a repository of all said at least one process instances is available for queries and retrieval by the runtime engine while executing said models or by external applications.

Claim **21**. (currently amended) The method ~~A computer program product (code generator) comprising a computer usable medium having computer readable code embodied therein for execution on a general purpose computer, said computer usable medium storing instructions that, when executed by the computer, cause the computer to generate the code of a software program implementing the applications defined in the visualizable computer executable modeling language system~~ of claim **2**, further comprising:

~~wherein: a plurality of display elements for displaying screen objects are displayed on a display screen and components are displayed corresponding to said screen objects;~~

defining each of the applications ~~is defined~~ by a single said process model and a hierarchy of its sub-models; and

~~the code generator produces~~ generating code in a general purpose programming language implementing the application exactly as defined by said single process model and said hierarchy of its sub-models, thus eliminating the need for writing computer source code in any programming language to implement the application[[,]]

~~wherein the computer executes the application model that has been created~~.

Claim **22**. (Currently Amended) The method ~~software program~~ of claim **21**, wherein said general purpose programming language is [[Java]]JAVA.

Claim **23**. (Currently Amended) The method ~~software program~~ of claim **21**, wherein said general purpose programming language is C++.

Claim **24**. (currently amended) A method for overcoming the need to write computer source code in order to develop software applications, comprising:

creating models of the applications in a visualizable computer executable modeling language system, using a visual modeling tool, comprising:

defining each of said software applications as a hierarchy of process models, input and output slots, data models and flow rules[[;]], wherein the components and screen objects are used for defining each of the software applications, the defining step further comprising:

classifying some of said process models and said data models that must not contain any sub-process as atomic;

classifying all other process models and data models that contain one or more sub processes as composite;

defining each of said composite process models as a construction of ~~at least one of~~ one or more sub process models, said input and output slots, said data models, and said flow rules;

classifying inputs sent to the input slots into mandatory and optional;

defining a portion of said input slots ~~of~~ as being one of ~~the following sub-classifications:~~ mandatory[[;]] and optional, wherein a process that requires an input at each of its mandatory input slots for its initiation does not start before all said mandatory inputs are received;

defining each of said composite data models as a construction of ~~at least~~ one or more sub data models; [[and]]

defining each of said flow rules [[as]] by connecting a pair ~~of~~ selected from said input and output slots, said data models, and the one or more sub data models, wherein said flow rules define both data flow and process flow in the process and data models; and

automatically executing the logic defined by said created models without requiring further coding, wherein said process models , [[and]] data models, [[and]] input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules.

Claim **25**. (Currently Amended) The method of claim **24**, wherein the execution of the logic defined by said models is made by a runtime engine ~~dedicated computer program (runtime engine)~~.

Claim **26**. (Currently Amended) The method of claim **24**, wherein the implementation of the logic defined by said models is made by the source code ~~of a software program~~ in a general purpose programming language, which is generated by a code generator ~~dedicated computer program (code generator)~~.

Claim **27**. (currently amended) A software development platform having a processor for overcoming the need to write computer source code in order to develop software applications, comprising:

a visualizable computer executable modeling language for the definition of software applications ~~solutions~~, said definition developed by performing the steps of ~~comprising~~:

defining each of said software applications [[as]]in a hierarchy of process models, input and output slots, data models, and flow rules[[;]], wherein said process models, input and output slots, data models and flow rules are used for defining each of the software applications, the defining step further comprising:

classifying some of said process models and said data models that must not contain any sub-process as atomic;

classifying all other process models and data models that contain one or more sub processes as composite;

defining each of said composite process models as a construction of ~~at least one of~~ one or more sub process models, said input and output slots, said data models, and said flow rules;

classifying inputs sent to the input slots into mandatory and optional;

defining a portion of said input slots ~~of~~ as ~~having~~ being one of ~~the following sub-classifications:~~ mandatory[[;]] and optional, wherein a process that requires an input at each of its mandatory input slots for its initiation does not start before all said mandatory inputs are received;

defining each of said composite data models as a construction of ~~at least~~ one or more sub data models; and

defining each of said flow rules [[as]] by connecting a pair ~~of~~ selected from said input and output slots, said data models, and the one or more sub data models, wherein said flow rules define both data flow and process flow in the process and data models;

a visual modeling tool for defining said software applications ~~solutions~~ by at least one user as said hierarchies of models in said modeling language; and

a dedicated computer program to automatically execute said software applications without requiring further coding according to the logic defined by said hierarchies of models, wherein said process models, [[and]] data models, [[and]] input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules.

Claim 28. (Currently amended) The software development platform of claim 27, wherein said dedicated computer program is a runtime engine that automatically executes said software applications ~~solutions~~ at runtime, according to the logic defined by said hierarchies of models.

Claim 29. (Currently amended) The software development platform of claim 27, wherein said dedicated computer program is a code generator that automatically generates the source code ~~of a software program~~ in a general purpose programming language implementing the logic defined by said hierarchies of models.

Claim 30. (currently amended) A computer program product comprising ~~a computer usable medium~~ memory having computer readable code embodied therein for execution on a ~~general purpose~~ computer, said ~~computer usable medium~~ memory storing instructions that, when executed by the computer, cause the computer to perform a modeling method for defining software applications using a visualizable computer executable modeling language, said method comprising:

providing a plurality of display elements for displaying screen objects on a display screen;

displaying components corresponding to said screen objects [[;]] to create an application model using the modeling language by:

defining each of the software applications as a hierarchy of process models, input and output slots, data models, and flow rules[[;]], wherein the components and screen objects are used for defining each of the software applications, the defining step further comprising:

classifying some of said process models and said data models that must not contain any sub process as atomic;

classifying all other process models and data models that contain one or more sub processes as composite;

defining each of said composite process models as a construction of ~~at least one of~~ one or more sub process models, said input and output slots, said data models, and said flow rules;

classifying inputs sent to the input slots into mandatory and optional;

defining a portion of said input slots ~~of~~ as ~~having~~ being one of ~~the following sub-classifications:~~ mandatory[[;]] and optional, wherein a process that requires an input at each of its mandatory input slots for its initiation, does not start before all said mandatory inputs are received;

defining each of said composite data models as a construction of ~~at least~~ one or more sub data models; and

defining each of said flow rules [[as]] by connecting a pair ~~of~~ selected from said input and output slots, said data models, and the one or more sub data models, wherein said flow rules define both data flow and process flow in the process and data models,

wherein said process models, [[and]] data models, [[and]] input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules[[,]];

~~such that once the software application is visually defined by the developer using the modeling language, a computer executes the application model that has been created using the computer with an appropriate display screen without requiring further coding.~~

thus creating the application model as visually defined on the display screen by the developer using the modeling language; and

enabling a computer to automatically execute the application model defined in said modeling language without requiring further coding according to the definition of the process models, data models, input and output slots, and flow rules arranged in the hierarchy.

Examiner's Statement of Reason(s) for Allowance

2. Claims 1-30 are allowed.

3. The following is an examiner's statement of reason s for allowance:

The prior arts of record, taken alone or in combination, fail to teach or fairly suggest at least:

said process models, data models, input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules; thus creating the application model as visually defined on the display screen by the developer using the modeling language; and enabling a computer to automatically execute the application model defined in said modeling language without requiring further coding according to the definition of the process models, data models, input and output slots, and flow rules arranged in the hierarchy as recited in the independent claims 1 and 30.

defining each of said flow rules by connecting a pair selected from said input and output slots, said data models, and the one or more sub data models, wherein said flow rules define both data flow and process flow in the process and data models; and automatically executing the logic defined by said created models without requiring further coding, wherein said process models , data models, input and output slots, and flow rules are arranged in a structural hierarchy conforming to a set of rigid composition rules as recited in claims 24 and 27.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Any inquiry concerning this communication or earlier communications from the examiner should be directed to INSUN KANG whose telephone number is (571)272-3724. The examiner can normally be reached on M-R 7:30-6 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis A. Bullock, Jr. can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Insun Kang/
Primary Examiner, Art Unit 2193